View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by NASA Technical Reports Server

N91-21094

Recent Update of the RPLUS2D/3D Codes

Y-L Peter Tsai
Sverdrup Technology, Inc.
NASA Lewis Research Center Group
Cleveland, Ohio 44135

NAS3-25266

Abstract

The development of the RPLUS2D/3D codes is summarized. These codes utilize LU algorithms to solve chemical non-equilibrium flows in a body-fitted coordinate system. Recent improvements include vectorization method, blocking algorithm for geometric flexibility, out-of-core storage for large-size problems, and an LU-SW/UP combination for CPU-time efficiency and solution quality.

I. Introduction

The technology for hypersonic vehicles has been developed for decades. Back in the sixties, the Air Force and NASA conducted the research on scramjet engines which were proposed for the Aero-Space Plane [1]. This was the prelude of the recent NASP (National Aero-Space Plane) program. Due to lack of powerful computers, early hypersonic scientists relied heavily on experiments. Analysis had to be done on a simplified basis. In the seventies, research on hypersonic planes and scramjet engines became dormant for more than a decade because of the Apollo and Shuttle missions. Recent hypersonic research in the eighties revived in a dramatically different environment in which the capability of computers had grown exponentially since their invention. The methodology in solving flow problems has also advanced significantly, thanks to the research of numerous mathematicians and fluid dynamicists. Three-dimensional, Reynolds-average Navier-Stokes simulations have thus been made feasible. As a result, CFD (Computational Fluid Dynamics) becomes a popular and promising tool for flow-prediction.

As a participant of the NASP program, NASA Lewis Research Center supported the development of the RPLUS2D/3D codes. This task is motivated by the need to numerically predict chemical non-equilibrium flows for the NASP program. Chemical reaction is an important factor for NASP-related flows. For the scramjet combustor, combustion is undoubtedly the vital issue. In the expansion nozzle, recombination of atoms and radicals generated upstream of the nozzle, and possibly some continuous combustion may account for a significant percentage of the thrust. At the NASP surface and inlet where the hypersonic airstream and solid body come in contact, and across the shock waves where the flow speed drastically changes, air-dissociation or even ionization may play important roles. The assumption of chemical-equilibrium may be considered to simplify the problems. However, the chemical-equilibrium assumption may result in poor prediction, thus its application is quite limited. For instance, in an H_2/air combustor, the chemical-equilibrium assumption produces an excessive amount of water vapor, which in turn over-predicts the performance of the combustor. The RPLUS2D/3D codes therefore incorporate chemical non-equilibrium models.

The size of the systems of equations for non-equilibrium chemistry imposes difficulties in obtaining the flow solution. The governing equations include the basic flow equations (continuity, Navier-Stokes, and energy equations) in addition to a number of species transport equations. It is not uncommon for a chemistry model to consider more than ten species, which require the same number of species equations. Multi-dimensional solution of such systems is a major task. It was not until recent advances of computer capability that three-dimensional simulation of chemically non-equilibrium flows has become feasible. Nevertheless, the efficiency of the numerical scheme remains

a vital issue.

One of the important features of the RPLUS2D/3D codes is the use of LU algorithms to march in time. In the past, both explicit and implicit algorithms for time-integration have been used for calculating chemical non-equilibrium flows. Among explicit algorithms, the MacCormack [2-5] and Runge-Kutta schemes [4,6] are most notable. The SPARK code [2] developed at NASA Langley center adopts the MacCormack scheme, and more recently, attempted a higher order MacCormack scheme with some success [3]. Shuen and Liou [6] utilized a two-stage Runge-Kutta in conjunction with spatially upwind-differenced method. To alleviate the restrictive limitation on time-step due to stiff source terms, Bussing and Murman [4] proposed implicit treatment for the source terms in these explicit methods. A number of other researchers prefer marching in time implicitly. Candler and MacCormack [7] implemented an implicit Gauss-Siedel line-relaxation technique for ionized flows. Molvik and Merkle [8-9] utilized a three-dimensional AF (Approximate-Factorization) in the inner iteration of a dual time-step procedure specifically for time-accurate calculation. Walters, et al. [10] included in their GASP (General Aerodynamic Simulation Program) code what they called AF/relaxation and LU/relaxation algorithms. Lee and Deiwert [11] extended the implicit flux-vector splitting algorithm of F3D [12] to include non-equilibrium chemistry. In general, implicit time-marching provides more temporal damping than explicit time-marching, and potentially gives better overall efficiency. However, implicit methods must be implemented with caution. The stability characteristics of some approximate factorizations depend strongly on the number of dimensions. An example is the celebrated ADI scheme [13] which has a rather restrictive CFL limit in three dimensions, due to the approximate-factorization error of the order Δt^3. The LU algorithms used by the RPLUS codes are two-factored, regardless of the number of dimensions. The convergence characteristics are therefore similar in one, two, or three dimensions.

The LU algorithm originally adopted by the RPLUS code is the so-called LU-SSOR (Symmetric Successive Over-Relaxation) or LU-SGS (Symmetric Gauss-Siedel) proposed by Yoon and Jameson [14-15]. This scheme was implemented by Shuen and Yoon [16] in the RPLUS2D code, and later extended to RPLUS3D by Yu, Tsai, and Shuen [17]. The implicit operator of the LU-SSOR is constructed by upwind differencing the specially formulated split flux Jacobians. For non-reacting flows, the resulting implicit operator avoids the need for matrix inversion, and is particularly suitable for solving large systems of equations. Even for reacting flows where source terms are treated implicitly, the requirement for matrix inversion is kept at a minimum. Recently, Tsai and Hsieh [18] modified the split flux Jacobians based on a similarity transformation to construct the so-called LU-SW (Steger-Warming). Along with the LU-SW, the right-hand-side calculation was modified from the original central-differencing to the upwind-differencing with Van Leer's flux-vector splitting. The LU-SW does require block-matrix inversion; however, the LU-SW/UP (UPwind-differencing) combination provides much stronger temporal damping. Tsai and Hsieh showed the LU-SW/UP can be more efficient in terms of CPU time than the original LU-SSOR/CD (Central-Differencing) in two dimensions. Since both LU algorithms remain two-factored in any number of dimensions, similar improvement can be expected in three dimensions.

This paper gives an updated summary of the RPLUS development. Besides the algorithm, improvements are made on the capability and flexibility in real application. Most notable improvements are the use of a blocking algorithm and the use of out-of-core storage. The blocking algorithm tremendously broadens the geometry to which the RPLUS codes can be applied. Difficulty in grid generation is also alleviated by the blocking capability. For large scale problems where the required memory storage exceeds the resources available, out-of-core storage is a necessary compromise. This has recently been made available for the RPLUS codes. These new features are addressed in detail.

II. Governing Equations

The governing equations solved by the RPLUS codes are the compressible, Reynolds-averaged Navier-Stokes equations and species transport equations. For completeness, the three-dimensional

equations are formulated. Written in a strong conservative form, the governing equations can be expressed as follows:

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}(E - E_v) + \frac{\partial}{\partial y}(F - F_v) + \frac{\partial}{\partial z}(G - G_v) = H \tag{1}$$

Here x, y, and z are Cartesian coordinates, Q is the dependent variable, E, F, and G are the convective flux vectors:

$$Q = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e \\ \rho Y_i \end{pmatrix}, \qquad E = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(\rho e + p) \\ \rho u Y_i \end{pmatrix} \tag{2}$$

$$F = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v(\rho e + p) \\ \rho v Y_i \end{pmatrix}, \qquad G = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w(\rho e + p) \\ \rho w Y_i \end{pmatrix}$$

E_v, F_v, and G_v are the viscous flux vectors:

$$E_v = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - q_x \\ -\rho \hat{u}_i Y_i \end{pmatrix}$$

$$F_v = \begin{pmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ u\tau_{yx} + v\tau_{yy}v + w\tau_{yz} - q_y \\ -\rho \hat{v}_i Y_i \end{pmatrix} \tag{3}$$

$$G_v = \begin{pmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ u\tau_{zx} + v\tau_{zy}v + w\tau_{zz} - q_z \\ -\rho \hat{w}_i Y_i \end{pmatrix}$$

and H is the source vector:

$$H = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \dot{\omega}_i \end{pmatrix} \tag{4}$$

The specific total energy, e, shear stress components, and heat flux components are given as

$$e = \sum_{i=1}^{N_s} e_i + \frac{1}{2}(u^2 + v^2 + w^2) \tag{5}$$

$$\tau_{xx} = 2\mu\frac{\partial u}{\partial x} - \frac{2}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right) \tag{6.a}$$

$$\tau_{yy} = 2\mu\frac{\partial v}{\partial y} - \frac{2}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right) \tag{6.b}$$

$$\tau_{zz} = 2\mu\frac{\partial w}{\partial z} - \frac{2}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right) \tag{6.c}$$

$$\tau_{xy} = \tau_{yx} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \tag{7.a}$$

$$\tau_{yz} = \tau_{zy} = \mu\left(\frac{\partial v}{\partial y} + \frac{\partial w}{\partial y}\right) \tag{7.b}$$

$$\tau_{zx} = \tau_{xz} = \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \tag{7.c}$$

$$q_x = -k\frac{\partial T}{\partial x} + \rho\sum_{i=1}^{N_s} h_i Y_i \hat{u}_i \tag{8.a}$$

$$q_y = -k\frac{\partial T}{\partial y} + \rho\sum_{i-1}^{N_s} h_i Y_i \hat{v}_i \tag{8.b}$$

$$q_z = -k\frac{\partial T}{\partial z} + \rho\sum_{i-1}^{N_s} h_i Y_i \hat{w}_i \tag{8.c}$$

In the preceding expressions, ρ is the density, u, v, and w are the Cartesian velocity components, p is the pressure, and e is the specific total energy. The subscript i identifies each species, and N_s is the total number of species. For ith species, Y_i, e_i, h_i, and $\dot{\omega}_i$ are its mass fraction, specific internal energy, enthalpy, and production rate, respectively. The enthalpy of species i is obtained by an integration of C_p versus temperature:

$$h_i = \int_o^T C_{pi} dT \tag{9}$$

where C_{pi} is the constant pressure specific heat which is expressed as a fourth order polynomial of temperature:

$$C_{pi} = C_{pi0} + C_{pi1}T + C_{pi2}T^2 + C_{pi3}T^3 + C_{pi4}T^4 \tag{10}$$

The internal energy of species i can be obtained from h_i using the ideal gas assumption which is valid for high temperature:

$$e_i = h_i - R_i T \tag{11}$$

where R_i is the gas constant for species i. The diffusion velocity components, $\hat{u}_i$, $\hat{v}_i$, and $\hat{w}_i$ are calculated by Fick's law [23]:

$$Y_i \hat{u}_i = -D_{im}\frac{\partial Y_i}{\partial x} \tag{12.a}$$

$$Y_i \hat{v}_i = -D_{im} \frac{\partial Y_i}{\partial y} \tag{12.b}$$

and

$$Y_i \hat{w}_i = -D_{im} \frac{\partial Y_i}{\partial z} \tag{12.c}$$

where

$$D_{im} = (1 - X_i) / \sum_{i \neq j} \frac{X_i}{D_{ij}} \tag{13}$$

is the effective binary diffusivity of species i in the gas mixture, and X_i is the mole fraction of species i. Diffusive properties such as viscosity and thermal conductivity are considered as polynomials of temperature, and the diffusive properties of the mixture are calculated based on Wilke's mixing rule [24,25]. The binary mass diffusivities are calculated using Chapman-Enskog theory in conjunction with Lennard-Jones intermolecular potential functions. [25]

The numerical solution of Eq. (1) is performed in a general, body-fitted coordinate system, (ξ, η, ζ). Coordinate transformation of Eq. (1) gives

$$\frac{\partial \hat{Q}}{\partial \tau} + \frac{\partial}{\partial \xi}\left(\hat{E} - \hat{E}_v\right) + \frac{\partial}{\partial \eta}\left(\hat{F} - \hat{F}_v\right) + \frac{\partial}{\partial \zeta}\left(\hat{G} - \hat{G}_v\right) = \hat{H} \tag{14}$$

where

$$\hat{Q} = hQ \tag{15.a}$$
$$\hat{E} = h(\xi_x E + \xi_y F + \xi_z G) \tag{15.b}$$
$$\hat{F} = h(\eta_x E + \eta_y F + \eta_z G) \tag{15.c}$$
$$\hat{G} = h(\zeta_x E + \zeta_y F + \zeta_z G) \tag{15.d}$$
$$\hat{E}_v = h(\xi_x E_v + \xi_y F_v + \xi_z G_v) \tag{15.e}$$
$$\hat{F}_v = h(\eta_x E_v + \eta_y F_v + \eta_z G_v) \tag{15.f}$$
$$\hat{G}_v = h(\zeta_x E_v + \zeta_y F_v + \zeta_z G_v) \tag{15.g}$$
$$\hat{H} = hH \tag{15.h}$$

in which h is the cell volume.

III. Recent Advances

The RPLUS2D code was completed in 1987 and extended to RPLUS3D in 1988. During this period of time, the codes remained research-oriented and modification was constantly made for different problems. They were hardly user-friendly due to the lack of user interfaces. From 1989 to present, a series of improvements have been made to render the RPLUS codes user-oriented. Parallel to these efforts, new algorithms for improving efficiency and solution quality were also explored. The details are addressed below.

Program Vectorization

Before a code can be vectorized, the programmer first identifies the operations for which parallel processing is possible. Once this is done, vectorization usually can be achieved with proficiency in programming. In cases when parallel processing is not possible, special hardware is required to accomplish vectorization. The implicit operator of the LU scheme has a recursive property. In the

LU scheme, the ΔQ of the point (i,j,k) requires updated ΔQ at points $(i-1,j,k)$, $(i,j-1,k)$, and $(i,j,k-1)$ in the Lower sweep, and updated values at $(i+1,j,k)$, $(i,j+1,k)$, and $(i,j,k+1)$ in the Upper sweep. At first glance, parallel processing seems to be out of the question. However, the solutions of the points in a plane normal to the sweeping direction are independent and can be parallel-processed. In other words, the points on the planes represented by the equation, $i+j+k=$ constant can be parallel-processed. The schematic of these planes is shown in Fig. 1.

The vectorization of the program is done by reorganizing the indices of the grid points for parallel-processing planes. Before the main iteration begins, mappings between the three-dimensional index, (i,j,k), and two sets of two-dimensional indices, $(ipoint, iplane)$, are generated for the Lower and Upper sweeps. The index *ipoint* identifies the points in a parallel-processing plane, and *iplane* identifies the planes. These mappings are stored as sets of integer arrays which can be invoked by the Lower and Upper sweeps. On CRAY computers, the mapping arrays can be used directly in DO-loops to achieve efficient vectorization. The strategy described does have short-comings. The rather short vector-length near the upper and lower corners makes vectorization somewhat incomplete. Nevertheless, these inefficient vector-processing regions do not constitute a significant efficiency-reduction on the CRAY for two reasons. Firstly, the vector length grows rapidly away from the corners. (The vector length is approximately proportional to square of the distance between the plane and the corner.) Secondly, CRAY's maximum computation speed can almost be reached with a rather short vector-length (e. g., 100). In fact, the CPU time for the implicit operator is reduced by a factor of nearly 10 simply by the described strategy.

The vectorization of the right-hand-side is relatively straightforward. In principle, three-dimensional vectorization is possible for the RHS. However, our experience on CRAY shows very little difference among one-, two-, and three-dimensional vectorization due to its architecture. The program is therefore vectorized in one dimension for better readability of the codes.

Blocking Logic

In order to enhance the capability of the RPLUS codes in handling complex geometries and flows, a blocking algorithm is implemented. The need for blocking is obvious when structured grids are used. Many geometries encountered in practical application, e. g. the nose-to-tail of the NASP, cannot be represented with a single-block grid. In association with the blocking algorithm, a set of flexible boundary conditions is installed, allowing specifying well-posed boundary conditions at grid-cell level. The boundary conditions currently installed include supersonic inflows and outflows, subsonic inflow, no-slip and tangency, freestreams, and interior boundaries (interfaces of blocks).

Blocking the grid does impose difficulty in using the LU algorithm. With the requirement for vectorization, the coding logic can be exceedingly complex. Currently the RPLUS codes use a simplified method in which the LU sweeps are performed independently for each block rather than performed for the whole grid. Figure 2 shows the schematic of this method for a two-dimensional, blocked grid. Notice that at the block interfaces, the boundary conditions for the LU sweeps are the same as any other boundaries. The instability this simplified method may incur is no more than what other boundaries can produce. Actual calculations also confirm that there is no apparent stability problem or deterioration of convergence speed on the interfaces.

The block-RPLUS codes have found many applications since their completion. One example is the study of combustion and mixing phenomenon of the dump combustor in a scramjet engine [19]. The geometry and flow configuration are shown in Fig. 3.a and 3.b. for H_2/air dump combustors with parallel and transverse injectors. The rearward facing step is a common gimmick for enhancing mixing. Grid generation for such geometry can be a major task if only one block is allowed to be used. With two blocks, on the other hand, the grid generation becomes exceedingly simple.

The hydrogen mass fraction and temperature distribution for both cases are shown in Fig. 4 and Fig. 5, respectively. The grid sizes are 41,000 points for the transverse-injection case and 23,700 points for the parallel-injection case. The chemistry model used is a 9-species, 18-step H_2/air reaction model by Brabbs [20]. Each case takes approximately 20 to 30 hours of CPU time on a CRAY-2. Figures 4.a and 4.b show the hydrogen mass fraction for transverse- and parallel-injection

cases. Comparing the downstream hydrogen mass fraction shows that the transverse-injection case provides a more complete combustion. This is due to a longer flow residence time and a better mixing mechanism of the transverse jet. The bending of the transverse jet results in a secondary flow which provides an excellent mixing mechanism. The secondary flow can be visualized in Fig. 4.a in which the transverse jet develops into a so-called kidney shape. The advantage of using the parallel jet is the smaller pressure loss as opposed to the transverse jet. The corresponding temperature distributions are shown in Figs. 5.a and 5.b. In both cases the recirculation zones generated by the steps have low subsonic speed and high temperature. The temperature distributions show that the mixing pattern in the parallel injection is somewhat different from that in the transverse injection. In the parallel injection case, the jets tend to recirculate sideways into the corner, while the transverse injection tends to mix the jets with the upstream air.

Out-of-Core Storage

While there is a growing demand for large-scale, three-dimensional computations, sufficient resources for core memory may not always be available. An alternative for such a dilemma is the use of secondary storage. Recently the possibility of utilizing secondary storage for the RPLUS3D code has been assessed. An experimental RPLUS version that requires a much smaller core memory has been developed. This version has been tested for a typically large grid of 360,000 ($100 \times 60 \times 60$) points. It requires only 5.7 Megawords for solving a nine-species chemical system, as opposed to 50 Megawords for the original version. For the secondary storage, the SSD (Solid-state Storage Device) currently available to CRAY-XMP and YMP appears to be the best candidate. Although using magnetic disks is also possible, the waiting time for the mechanical movement may result in a long lapse time for a small amount of CPU time.

The idea of out-of-core storage is stocking the major portion of data while operating on a minor portion of the data. For a three-dimensional problem at least two strategies can be considered. The first strategy is dividing the three-dimensional domain into much smaller subdomains. The data for each subdomain are stored in the secondary storage ordered on direct-access records. Operations are then done for one subdomain at a time, and updated data are written to the proper records when the operations are finished. The second strategy is storing the data based on two-dimensional planes. At any given time, data for a number of planes may be required to appear on the stage. The updated data are stored back to the designated records after required operations are done. The I/O efficiency of a strategy depends not only on the frequency of accessing but also on the manner that the data are stored. For direct-access storage, longer records give more efficient access. If both strategies give identical results, the first strategy seems to be more attractive since it gives the freedom of controlling the record-length. For implicit schemes, however, the first strategy requires using explicit boundary conditions on the interfaces of the subdomains. The RPLUS3D therefore chooses the second strategy.

The programming logic currently used by the RPLUS3D for storing the two-dimensional planes is by no means the most efficient one. Due to vectorization considerations, the planes whose data are stored in records are of constant-I, -J, -K for RHS calculation and of constant-I+J+K for LHS. In each iteration, the data files have to be reorganized four times to ensure efficient retrieve. This demands a tremendous amount of I/O. A more efficient way is to vectorize the RHS calculation for constant-I+J+K planes, the same as for the LHS calculation. This can avoid the need to reorganize the data files. The experimental version of RPLUS3D is currently modified to adopt this strategy.

LU-SW and Upwind-differencing

The RPLUS codes are originally designed to use LU-SSOR/CD. The efficiency and solution quality of this algorithm have recently been re-assessed, and other possible algorithms have been explored [6,18]. Among them, the LU-SW/UP has been identified as a promising algorithm. This algorithm has been added to the RPLUS2D.

The finite-difference equations for the LU-SSOR and LU-SW have the same generic form. In

two dimensions, both can be expressed by

$$\begin{aligned}
&\left[\mathbf{I}+\Delta t\left(D_\xi^-\hat{\mathbf{A}}^+ + D_\eta^-\hat{\mathbf{B}}^+ - \frac{\hat{\mathbf{A}}^-}{\Delta\xi} - \frac{\hat{\mathbf{B}}^-}{\Delta\eta} - \hat{\mathbf{D}}\right)\right] \\
&\left(\mathbf{I}+\frac{\Delta t}{\Delta\xi}(\hat{\mathbf{A}}^+ - \hat{\mathbf{A}}^-) + \frac{\Delta t}{\Delta\eta}(\hat{\mathbf{B}}^+ - \hat{\mathbf{B}}^-)\right)^{-1} \\
&\left[\mathbf{I}+\Delta t\left(D_\xi^+\hat{\mathbf{A}}^- + D_\eta^+\hat{\mathbf{B}}^- + \frac{\hat{\mathbf{A}}^+}{\Delta\xi} + \frac{\hat{\mathbf{B}}^+}{\Delta\eta}\right)\right]\Delta\hat{\mathbf{Q}} \\
&= \Delta t RHS
\end{aligned} \tag{16}$$

The two LU algorithms have different ways in constructing the split flux-Jacobians. For LU-SSOR, the split flux Jacobians are defined as

$$\hat{\mathbf{A}}^+ = 0.5(\hat{\mathbf{A}} + \gamma_{\hat{\mathbf{A}}}\mathbf{I}) \tag{17.a}$$

$$\hat{\mathbf{A}}^- = 0.5(\hat{\mathbf{A}} - \gamma_{\hat{\mathbf{A}}}\mathbf{I}) \tag{17.b}$$

$$\hat{\mathbf{B}}^+ = 0.5(\hat{\mathbf{B}} + \gamma_{\hat{\mathbf{B}}}\mathbf{I}) \tag{17.c}$$

$$\hat{\mathbf{B}}^- = 0.5(\hat{\mathbf{B}} - \gamma_{\hat{\mathbf{B}}}\mathbf{I}) \tag{17.d}$$

where $\gamma_{\hat{\mathbf{A}}}$ and $\gamma_{\hat{\mathbf{B}}}$ are greater than the spectral radii of the associated flux Jacobians :

$$\gamma_{\hat{\mathbf{A}}} \geq \max(|\lambda_{\hat{\mathbf{A}}}|) \tag{18.a}$$

$$\gamma_{\hat{\mathbf{B}}} \geq \max(|\lambda_{\hat{\mathbf{B}}}|) \tag{18.b}$$

For the LU-SW, the split flux Jacobians are defined as

$$\hat{\mathbf{A}}^+ = \hat{\mathbf{M}}_\xi \Lambda_\xi^+ \hat{\mathbf{M}}_\xi^{-1} \tag{19.a}$$

$$\hat{\mathbf{A}}^- = \hat{\mathbf{M}}_\xi \Lambda_\xi^- \hat{\mathbf{M}}_\xi^{-1} \tag{19.b}$$

$$\hat{\mathbf{B}}^+ = \hat{\mathbf{M}}_\eta \Lambda_\eta^+ \hat{\mathbf{M}}_\eta^{-1} \tag{19.c}$$

$$\hat{\mathbf{B}}^- = \hat{\mathbf{M}}_\eta \Lambda_\eta^- \hat{\mathbf{M}}_\eta^{-1} \tag{19.d}$$

in which Λ_ξ^+, etc., are the diagonal eigenvalue matrices, and $\hat{\mathbf{M}}_\xi$, etc., are the right eigenmatrices.

In the LU-SSOR, the flux Jacobians are split in such a fashion that the block-matrix inversion is avoided. The eigenvalues of the resulting split flux-Jacobians are inconsistent with the characteristic speeds of the flow. Because of this inconsistency, relatively slow convergence is usually observed using this scheme. The LU-SW, on the other hand, requires block-matrix inversion and each iteration is more expensive. However, because the LU-SW scheme uses split Jacobians whose eigenvalues are consistent with the characteristic speeds of the flow, it is apt to give a convergence rate faster than the LU-SSOR in terms of number of iterations.

The efficiency of the LU-SSOR and LU-SW schemes are strongly affected by the method of the right-hand-side discretization, which can be central or upwind. The flux-vector splitting by Van Leer [21] is selected for upwind-differencing. Higher order accuracy is achieved by the MUSCL procedure [22]. The two LU schemes and two methods for right-hand-side discretization result in four combinations, namely, LU-SSOR/CD, LU-SSOR/UP, LU-SW/CD, and LU-SW/UP. Among these combinations, the LU-SW/CD can be shown to be unstable by a stability analysis [18]. The efficiency of the other three combinations is tested by three cases : (1) 15° ramp, (2) 20° ramp with expansion, and (3) jet in crossflow.

Figures 6.a, 7.a, and 8.a show the geometry and flow conditions for the three test cases. In cases 1 and 2, a premixed H_2/air flow passes through a ramp. Chemical reaction is activated by

the high temperature induced by oblique shocks, boundary layers, and their interaction. Case 3 simulates the combustion process in a hydrogen combustor. In all three cases, the optimum CFL number is infinity for the LU-SSOR and approximately 1.5 for the LU-SW. The flows are assumed to be laminar. Figures 6.b, 7.b, and 8.b. show the corresponding convergence histories. The CPU seconds required by each scheme for reducing the residual by one order are tabulated in Table. 1. It clearly shows that the LU-SW/UP scheme is the most efficient among the three.

	Case 1	Case 2	Case 3
LU-SSOR/CD	675	-	1500
LU-SSOR/UP	477	1908	-
LU-SW/UP	225	1620	900

Table 1. CPU Seconds for Reducing Residual by One Order.

Other Improvements

The most updated versions of the RPLUS codes are designed to cope with a variety of flows. They are able to compute perfect-gas or real-gas, single-species or multiple-species, inviscid or viscous, non-reacting or reacting flows. All these options can be used conveniently without modifying the codes. For chemistry models, a 9-species, 18-step H_2/air combustion model is stored internally as one of the default options. Suitable chemistry models will be installed in the future. Any other chemistry models can be defined through an input file. In addition to the H_2/air combustion model, real-gas properties such as the specific heats, viscosity and thermal conductivity of a number of species are also stored internally for convenient usage. All these features make the RPLUS codes extremely flexible and useful.

IV. Conclusion

Continuous effort at NASA Lewis has improved the RPLUS codes significantly. The updated RPLUS codes have included a number of new features for practical application. To handle complex geometry and simplify grid generation, a blocking logic is incorporated. Out-of-core storage shows potential in calculating large size problems with limited core-memory. The LU-SW/UP algorithm improves the overall efficiency as well as solution quality. These improvements not only make the RPLUS codes more ready for use, but also impose an impact on their future development. Future work includes advance turbulence modelling such as the $k - \epsilon$ modelling and PDF (Probability Density Function) modelling. For geometry flexibility, an algorithm for multiple, mismatched grid with global conservation law will be explored.

Acknowledgement

This work was supported by NASA Lewis Research Center under contract NAS3-25266. The computer time is provided by the Numerical Aerodynamic Simulation Program (NAS). The support is gratefully acknowledged.

References

1. Rubert, K. F., "Aero-Space Plane," Joint Meeting of the Bumblebee Aerodynamic Composite Design and Propulsion Panels, Silver Spring, Maryland, November 15-16, 1961.

2. Drummond, J. P., Rogers, R. C., and Hussainni, M. Y., "A Detailed Numerical Model of

a Supersonic Reacting Mixing Layer," AIAA paper 86-1427, AIAA/ASME/SAE/ASEE 22nd Joint Propulsion Conference, 1986.

3. Uenishi, K., Rogers, R.C., and Northam, G.B., "Three Dimensional Computations of Transverse Hydrogen Jet Combustion in a Supersonic Airstream," AIAA Paper 87-0089, AIAA 25th Aerospace Sciences Meeting, Jan. 12-15, 1987, Reno, Nevada.

4. Bussing, T.R.A, and Murman, E.M., "A Finite Volume for the Calculation of Compressible Chemically Reacting Flows," AIAA Paper 85-0331, AIAA 23th Aerospace Sciences Meeting, Jan. 14-17, 1985, Reno, Nevada.

5. Carpenter, M. H., "Three-Dimensional Computations of Cross-Flow Injection and Combustion in a Supersonic Flow," AIAA Paper 89-1870, AIAA 20th Fluid Dynamics, Plasma Dynamics and Lasers Conference, Buffalo, NY, June 12-14, 1989.

6. Shuen, J.-S., Liou, M.-S., "Flux Splitting Algorithms for Two-Dimensional Viscous Flows with Finite-Rate Chemistry," AIAA Paper 89-0388, AIAA 27th Aerospace Sciences Meeting, Jan. 9-12, 1989, Reno, Nevada.

7. Candler, G. V., and MacCormack, R. W., "The Computation of Hypersonic Ionized Flows in Chemical and Thermal Non-Equilibrium," AIAA Paper 88-0511, AIAA 26th Aerospace Sciences Meeting, Jan. 11-14, 1988, Reno, Nevada.

8. Molvik, G. A., and Merkle, C. L., "A Set of Strongly Coupled, Upwind Algorithms for Computing Flows in Chemical Non-Equilibrium," AIAA Paper 89-0199, AIAA 27th Aerospace Sciences Meeting, Jan. 9-12, 1989, Reno, Nevada.

9. Molvik, G. A., "Computation of Viscous Blast Wave Solutions with an Upwind, Finite-Volume Method," AIAA Paper 87-1290, June 1987.

10. Walters, R. W., Cinnella, P., Slack, D. C., and Halt, D., "Characteristic-Based Algorithms for Flows in Thermo-Chemical Non-Equilibrium," AIAA Paper 90-0393, AIAA 28th Aerospace Sciences Meeting, Jan. 8-11, 1990, Reno, Nevada.

11. Lee, S.-H., and Deiwert, G. S., "Calculation of Non-equilibrium Hydrogen-Air Reactions with Implicit Flux Vector Splitting Method," AIAA Paper 89-1700, AIAA 24th Thermophysics Conference, June 12-14, 1989, Buffalo, NY.

12. Ying, S. X., "Three-Dimensional Implicit Approximately Factored Schemes for the Equations of Gasdynamics," Ph. D. Thesis, Stanford University, Jun. 1986.

13. Pulliam, T. H., and Steger, J. L., "Implicit Finite-Difference Simulations of Three Dimensional Compressible Flow," *AIAA Journal*, Vol. 18, 1980, pp. 159.

14. Yoon, S., and Jameson, A., "An LU-SSOR Scheme for the Euler and Navier-Stokes Equations," AIAA Paper 87-0600, Jan., 1987.

15. Jameson, A., and Yoon, S., "Lower-Upper Implicit Schemes with Multiple Grids for the Euler Equations," *AIAA Journal*, Vol. 25, No. 7, July 1987, pp. 929-935.

16. Shuen, J. S. and Yoon, S., "Numerical Study of Chemically Reacting Flows Using an LU Scheme," AIAA paper 88-0436, Jan., 1988.

17. Yu, S-T, Tsai, Y-L P., and Shuen, J-S, "Three-Dimensional Calculation of Supersonic Reacting Flows Using an LU Scheme," AIAA paper 89-0391, 27th Aerospace Sciences Meeting, Jan. 9-12, 1989, Reno, Nevada.

18. Tsai, Y.-L. P., and Hsieh, K.-C., "Comparative Study of Computational Efficiency of Two LU Schemes for Non-Equilibrium Reacting Flows," AIAA paper 90-0396, 28th Aerospace Sciences Meeting, Jan. 8-11, 1990, Reno, Nevada.

19. Tsai, Y-L P., Yu, S-T, "Further Development of the RPLUS3D Code," Paper 8, 7th National Aero-Space Plane Symposium, Cleveland, Ohio, Oct. 23-27, 1989.

20. Brabbs, T. A., personal communication.

21. Van Leer, B., "Flux-Vector Splitting for the Euler Equations," Lecture Notes in Physics, Vol. 170, 1982, pp. 507-512.

22. Van Leer, B., "Towards the Ultimate Conservative Difference Scheme V. A Second-Order Sequel to Gudonov's Method," *J. Comput. Phys.*, vol. 32, 1979, pp. 101-136.

23. Kuo, K. K., (1986). *Principles of Combustion*, p. 165. John Wiley & Sons, Inc., New York.

24. Wilke, C. R., "A Viscosity Equation for Gas Mixture," *J. Chem. Phys.*, Vol. 18, No. 4, Apr. 1950, p. 517.

25. Reid, R. C., Prausnitz, J. M., and Sherwood, T. K., (1977), *The Properties of Gases and Liquids*, 3rd Ed., McGraw-Hill, New York.

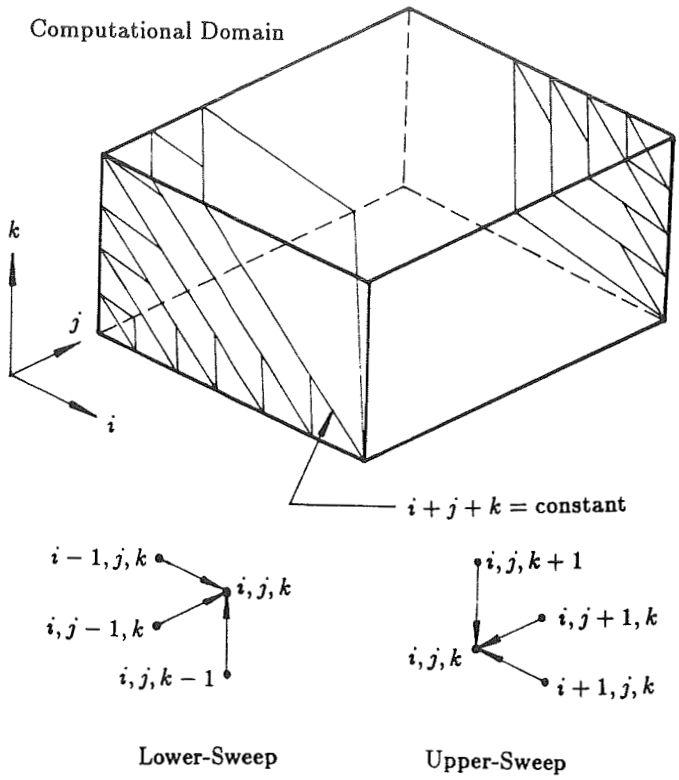

Fig. 1 Schematic of Program Vectorization.

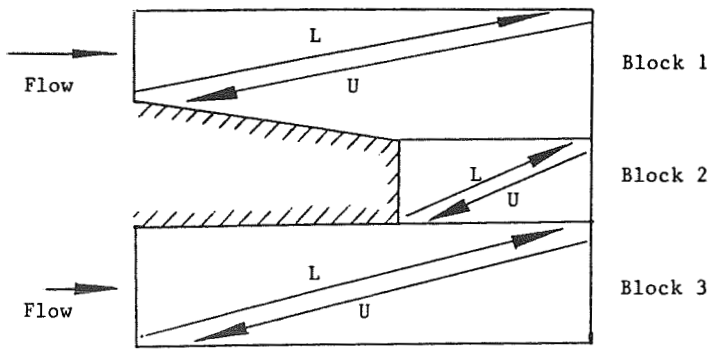

Fig. 2 Schematic of LU Sweeps in a Multi-Block Grid.

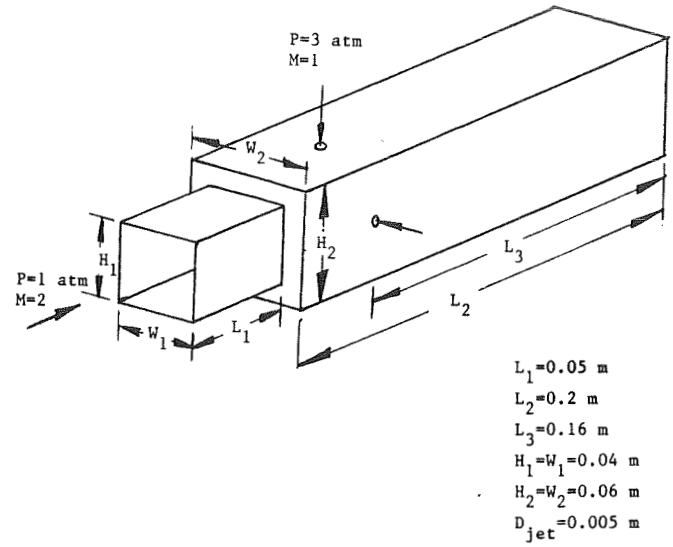

Fig. 3.a Geometry and Flow Conditions for Dump Combustor with Transverse Injection.

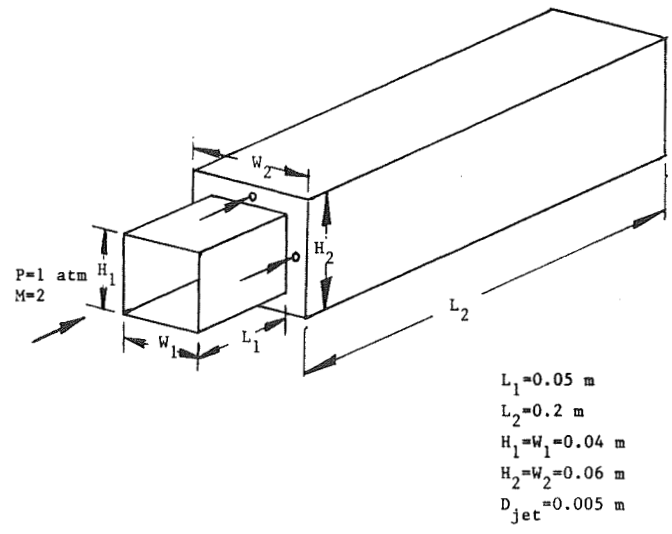

Fig. 3.b Geometry and Flow Conditions for Dump Combustor with Parallel Injection.

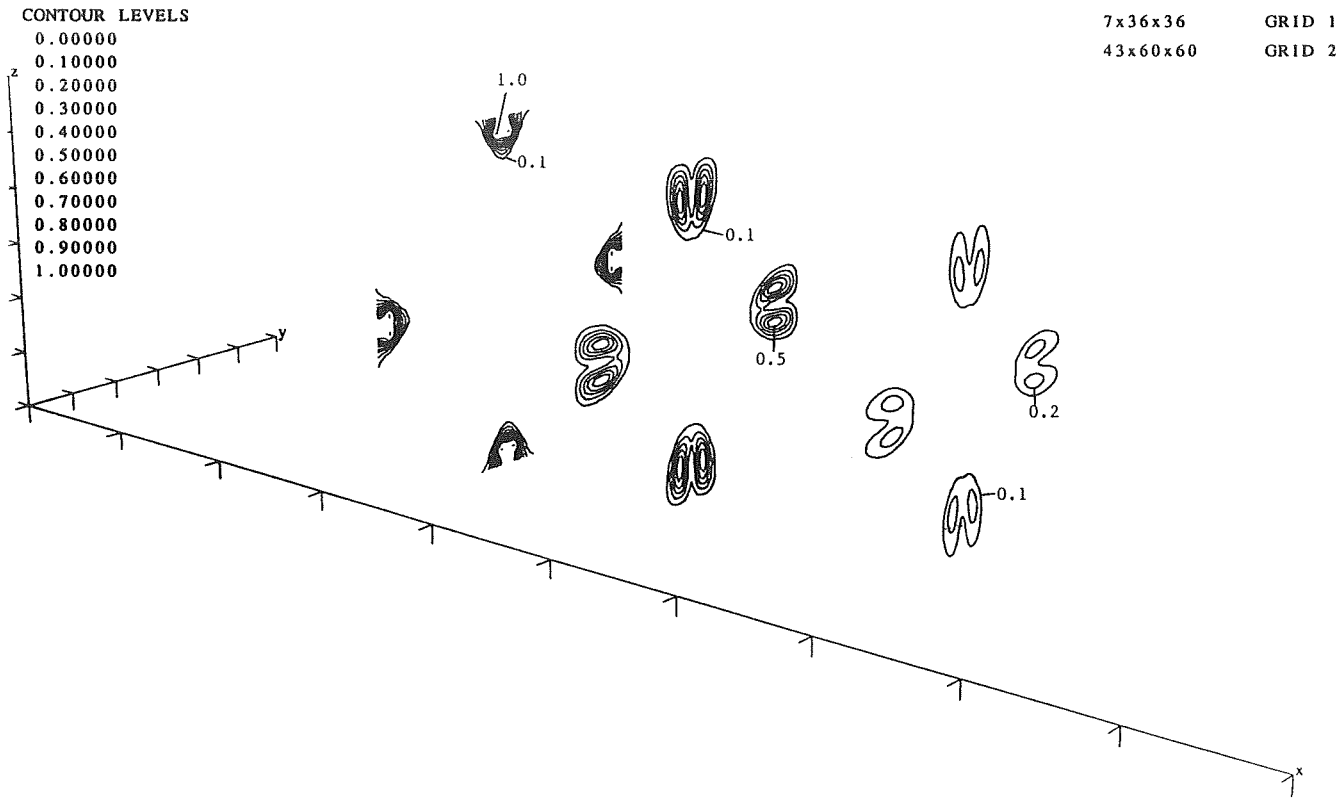

Fig. 4.a Hydrogen Mass Fraction for Dump Combustor with Transverse Injection.

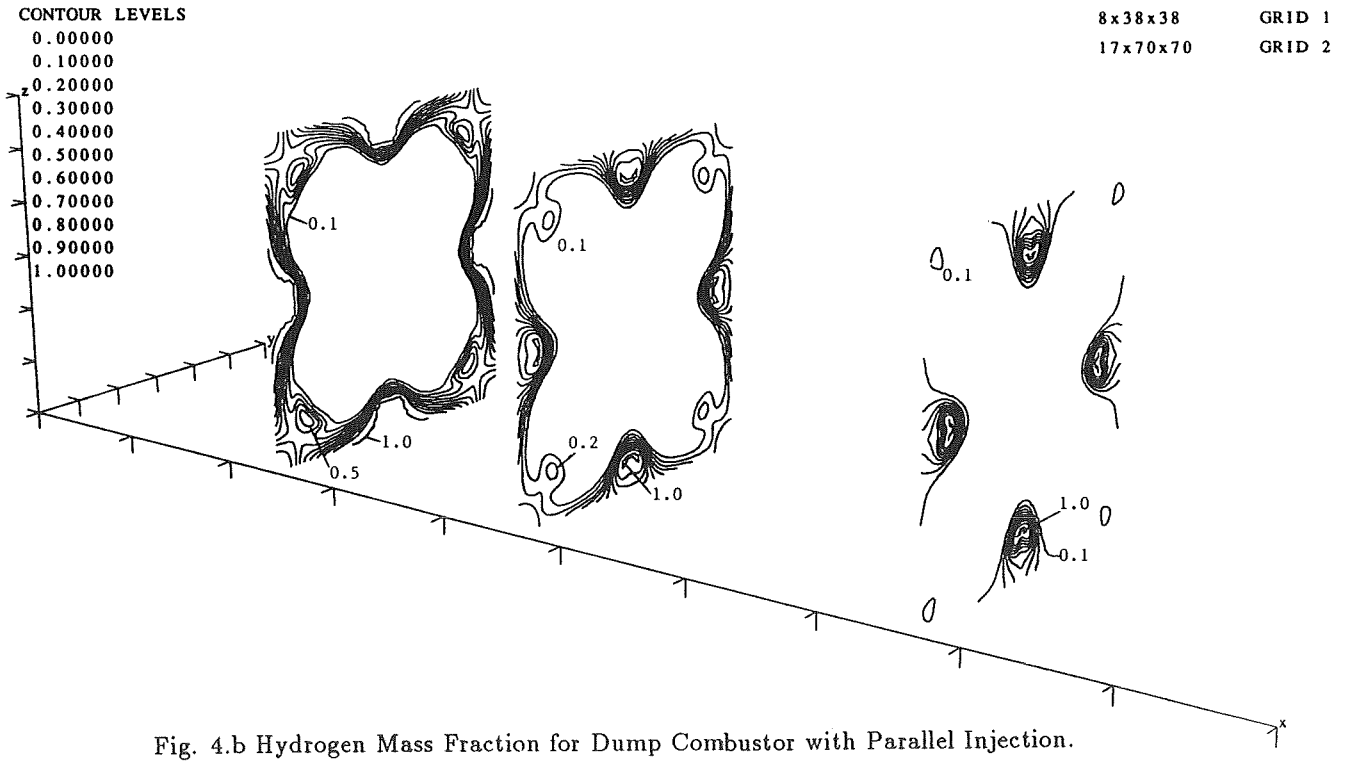

Fig. 4.b Hydrogen Mass Fraction for Dump Combustor with Parallel Injection.

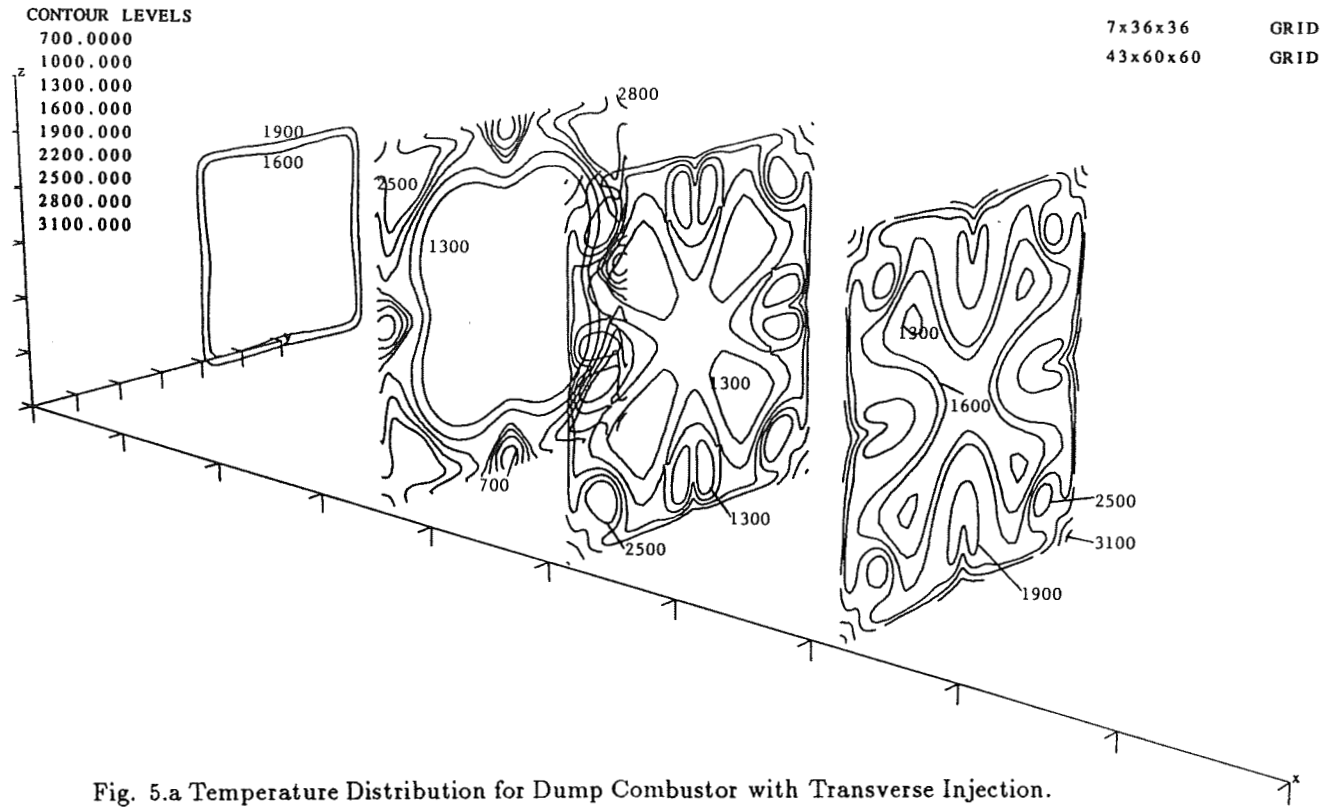

Fig. 5.a Temperature Distribution for Dump Combustor with Transverse Injection.

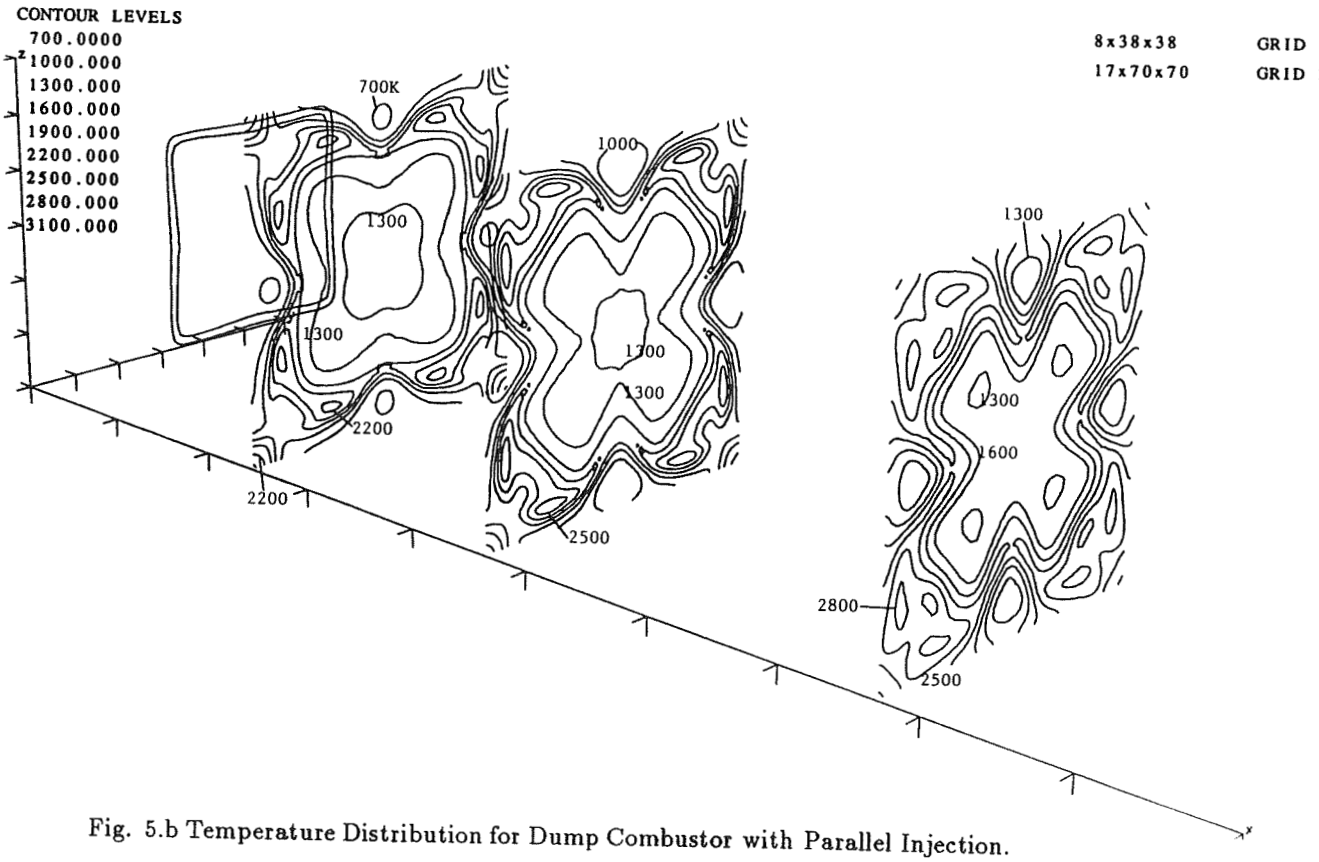

Fig. 5.b Temperature Distribution for Dump Combustor with Parallel Injection.

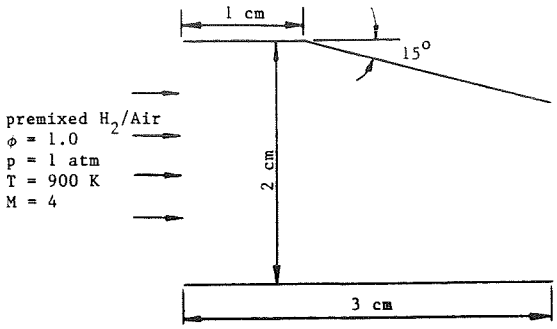

Fig. 6.a Geometry and Flow Conditions of 15° Ramp.

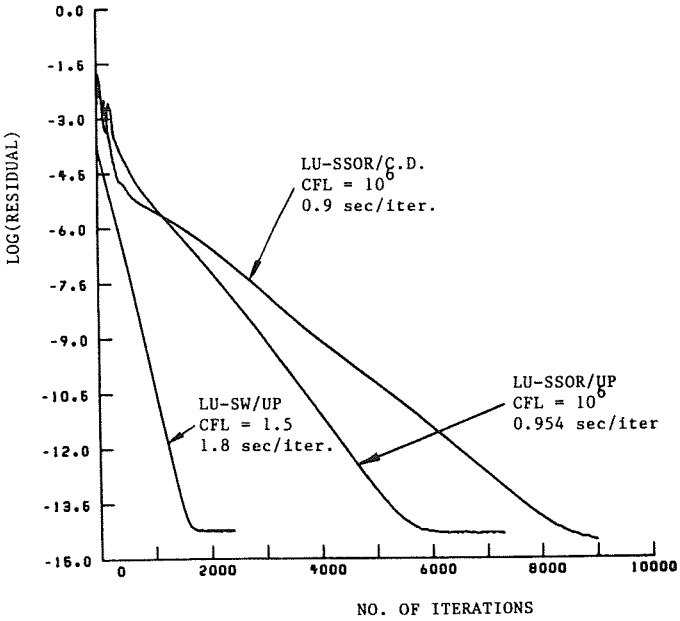

Fig. 6.b Convergence Rate for 15° Ramp.

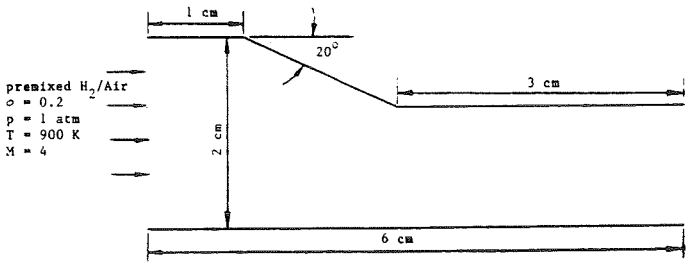

Fig. 7.a Geometry and Flow Conditions of 20° Ramp with expansion.

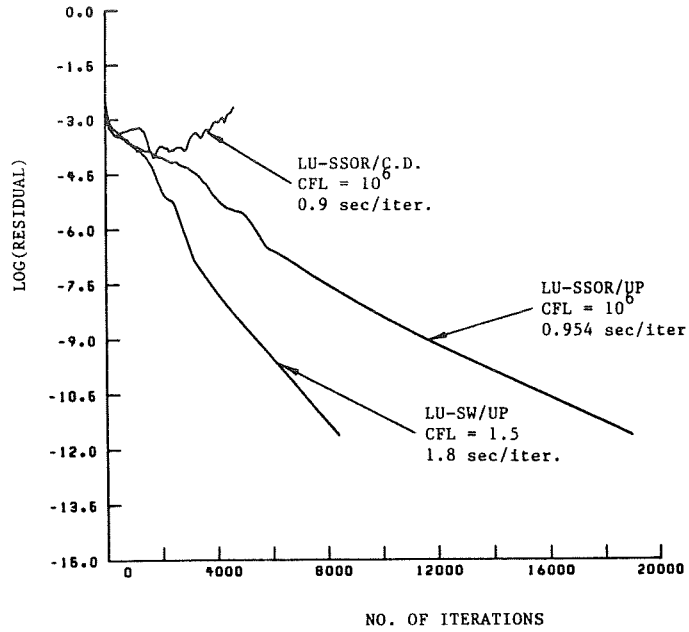

Fig. 7.b Convergence Rate for 20° Ramp with expansion.

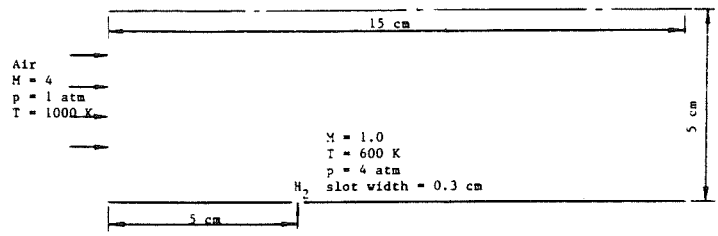

Fig. 8.a Geometry and Flow Conditions of Jet in Crossflow.

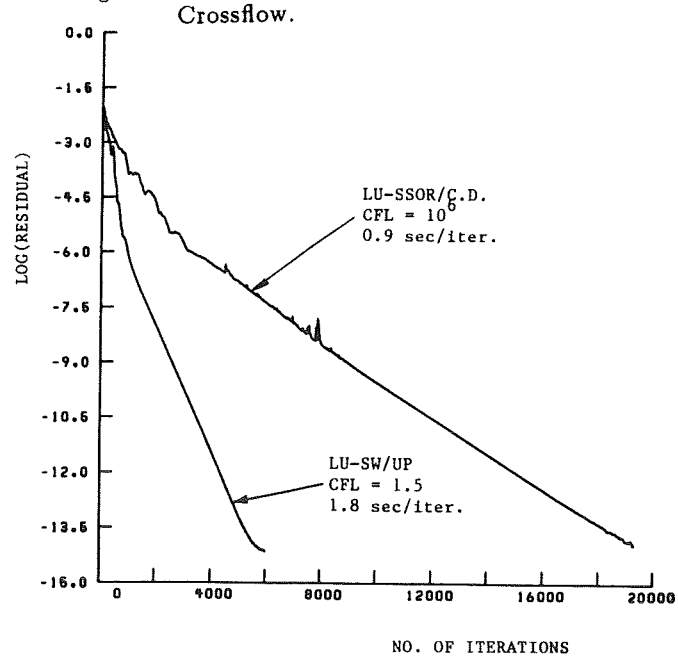

Fig. 8.b Convergence Rate for Jet in Crossflow.